Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 6 (2): 21-30, 2012
ISSN 1819-4311 / DOI: 10.3923/jse.2012.21.30
© 2012 Academic Journals Inc.

Research on Design Method of Dynamic Partial Reconfigurable System

Jiliang Zhang, Qiang Wu and Jiani Chen
College of Information Science and Engineering, Hunan University, Changsha, 410082, China

Corresponding Author: Jiliang Zhang, College of Information Science and Engineering, Hunan University, Changsha, 410082, China

ABSTRACT

In general, a large number of different Intellectual Property (IP) cores can be implemented on a System-On-Chip (SOC) in parallel. However, this was not resource efficient, as depending on the application, only a subset of those cores would active at the same time. This study focused on the design method of the Dynamic Partial Reconfigurable (DPR) system that design and implements a DPR system to address the problems. The result of experiment shows that DPR can greatly improve FPGA's resource utilization and save reconfigurable time.

Key words: Dynamic partial reconfigurable, field programmable gate arrays, design method, resource

INTRODUCTION

With the development of FPGA, the FPGA has become core processor component in the digital system and applications are increasingly widespread (Li *et al*., 2011; Peiravi and Rahimzadeh, 2009; Wu *et al*., 2009; Ramakrishnan and Shanmugavel, 2007). In recent years, the Xilinx has introduced reconfigurable technology that utilizes the programmable features of FPGA to time-share logic resources (Xilinx, 2004) so that the different logic circuits can be implemented in the FPGA sequentially and greatly improves the flexibility of FPGA. Reconfigurable technology has a large potential for widely use (Ramadass *et al*., 2007).

Reconfiguration is the process of changing the structure of a reconfigurable device at star-uptime, respectively at run-time (Compton and Hauck, 2002; Bobda, 2007). Reconfigurable technology is divided into static reconfiguration and dynamic reconfiguration and dynamic reconfiguration can be divided into global reconfiguration and partial reconfiguration. Dynamic reconfiguration allows the device portions that are not directly involved in the reconfiguration to run without interruption through the reconfiguration process (Corbetta *et al*., 2009). Partial Reconfiguration reduces the granularity of reconfiguration to be a set of columns or rectangular region of the device. Decreasing the granularity of reconfiguration results in reduced configuration filesizes and, thus, reduced configuration times. When compared to one bitstream of a non-partial reconfiguration implementation, smaller modules resulting in smaller bitstream filesizes allow an FPGA to implement many more hardware configurations with greater speed under similar storage requirements (Parris, 2008).

As long as the system designed meets one of the following questions, it can be resolved through DPR:

- System's resource requirement is greater than FPGA's available resources
- System designed has multiple mutually exclusive tasks
- Reduce system energy consumption
- System has temporarily no need to hardware resources

DPR has unique features which make it more and more promising, therefore, effective and fast design method for DPR system is very important instructive for designers (Claus *et al*., 2007). Currently, many researchers have studied the reconfigurable computing (Compton and Hauck, 2002; Bobda, 2007) and have derived a large number of design models and platforms and have applied the technology to the various fields including aerospace and military. However, the related research on the dynamic reconfiguration is relatively less and is still in an initial start-up phase, the design method of the DPR system has yet to be deepened. In study of Becker *et al*. (2007), the reconfigurable system which was on single FPGA chip with module-based method was for reusing resources to improve performance. Those references mentioned were all module-based design method and used the command line to achieve the reconfigurable system. This study focuses on the design method of the DPR system and finally we found that the method based on EAPR which combines with PlanAhead is able to complete the design of the DPR system more effectively than existing methods and approaches based on some comparisons. Therefore, we describe the design flow based on the above method. The design method proposed in this study; (1) can effectively shorten the design time (2) easy to modify the complex reconfigurable system repeatedly (3) improve the efficiency of design.

BASIC ARCHITECTURE OF DPR SYSTEM

The DPR system divides the FPGA into two regions: dynamic region (also called reconfigurable region) and static region. The logic functions which are in the dynamic region can be changed during the running time, while the other logic functions which are in the static region are not affected and can continue to run when the dynamic region is being configured (Papadimitriou *et al*., 2010). The dynamic region could contain more than one reconfigurable module and each one can achieve a logic circuit function, but it can be replaced by other reconfigurable modules to achieve new logic function. Communicate between static region and dynamic region is bus macro that is a fixed routing resource and is an important safeguard to achieve DPR system based on FPGA. Diagram of the basic structure of the DPR system is shown in Fig. 1.

THE DPR SYSTEM DESIGN AND IMPLEMENTATION

The DPR was first proposed by Xilinx. Xilinx's Virtex FPGA supported the DPR technology. There are mainly two design methods for DPR system: Difference-based and Module-based which were introduced in detail (Xilinx, 2004). And the defects which existed in the Difference-based method were introduced. EAPR (Xilinx, 2006) is similar to the Module-based method but there are some differences between them: (1) In the EAPR design method, the dynamic region allows for the arbitrary rectangular and allows the global signal of the static modules directly through this region without using the bus macro to connect the signal. (2) The Slice-based bus macro can replace the bus macro which is Look Up Table (LUT) based in the EAPR. This study studies on EAPR method for designing the DPR system and details its design flow to make users more clearly about how to design a DPR system effectively.

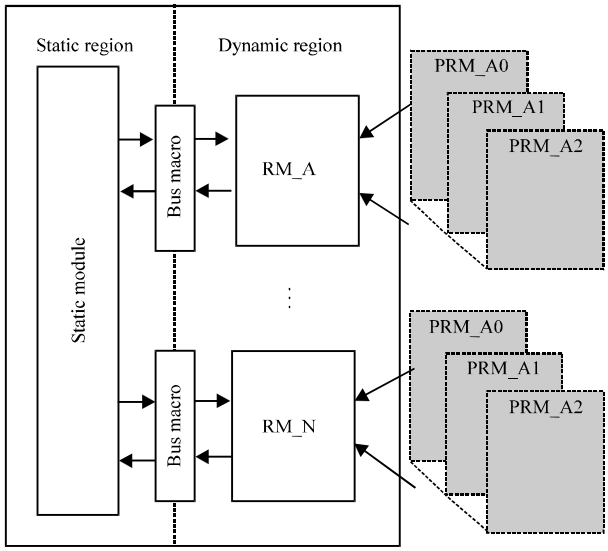

Fig. 1: Basic architecture of DPR system

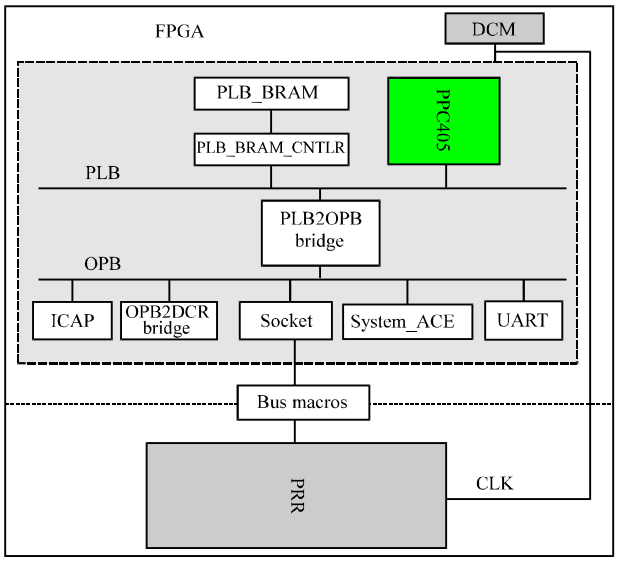

Fig. 2: A specific structure of a DPR system

The proposed design method with EAPR for the DPR system: The design flow of the DPR system is different from flow of the general FPGA and has more stringent restrictions and regulations. This paper will detail the design flow through a specific design in order to make designer have a more profound understanding the design approach with the help of the ISE9.1i, EDK9.1i and PlanAhead9.2.7.

Architecture diagram of the DPR system: This section will discuss a specific structure which deduces from the basic architecture (Fig. 1) as shown in Fig. 2. The static region is composed of static modules such as the PPC405 processor, UART, System ACE, ICAP and Socket and the system

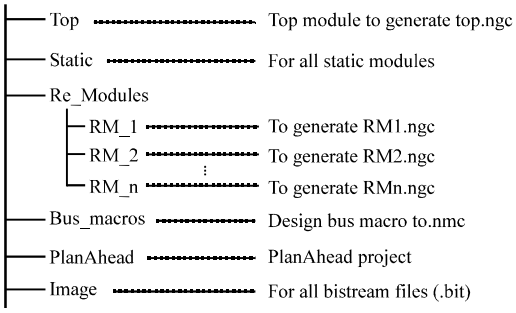

Fig. 3: File structure

only contains a reconfigurable module which can be replaced by other reconfigurable modules. PPC405 processor which controls the configurable flow, is the key to achieve dynamic reconfiguration and users reconfigure the FPGA's reconfigurable module through ICAP internal (Becker *et al.*, 2007; McDonald, 2008).

File structure: Because the design flow of the DPR system is complex and involves with lots of files, it is complicated to achieve. The appropriate file structure can make the design flow more clearly and simplify. Therefore, in order to facilitate the realization of the system, this study recommends the file structure that can organize and manage the design and implementation files and documents at all stages as shown in Fig. 3 and we can extend the file structure for different applications.

Design flow of the DPR system: The design flow of the DPR system generally is as design top module, design static module, design reconfigurable module, merging modules and as shown in Fig. 4.

Design top module: It is the main purpose that the top module is used to assign the overall layout of the FPGA according to the practical application and divide the FPGA into the static region and reconfigurable region and establish connection between static modules and reconfigurable modules. The static modules, reconfigurable modules and bus macros are all in the top module in the form of sub-module and the design of these modules are relatively independent. In the top module, these sub-modules are "black box", only providing a logical interface not implementing specific function. In addition, the global clock resources (DCM and BUFG) and IOB (input/output) are all defined in the top module. Design top module should follow these rules: Firstly, all the sub-modules port must be explicitly defined as input or output and cannot be defined as a buffer. Secondly, reconfigurable module requires using the global clock and cannot share any signals except global clock signal, including reset, enable signals and so on. Thirdly, it is necessary to minimize the number of reconfigurable module.

Design static module: Static module is the part whose logic functions do not change and continue to run when the system is reconfiguring. Static modules control and assist the system to complete the reconfiguration operation.

In the DPR systems designed for this study, the static modules are composed of the processor and a number of peripheral devices that the system needs which are selected by EDK's Platform

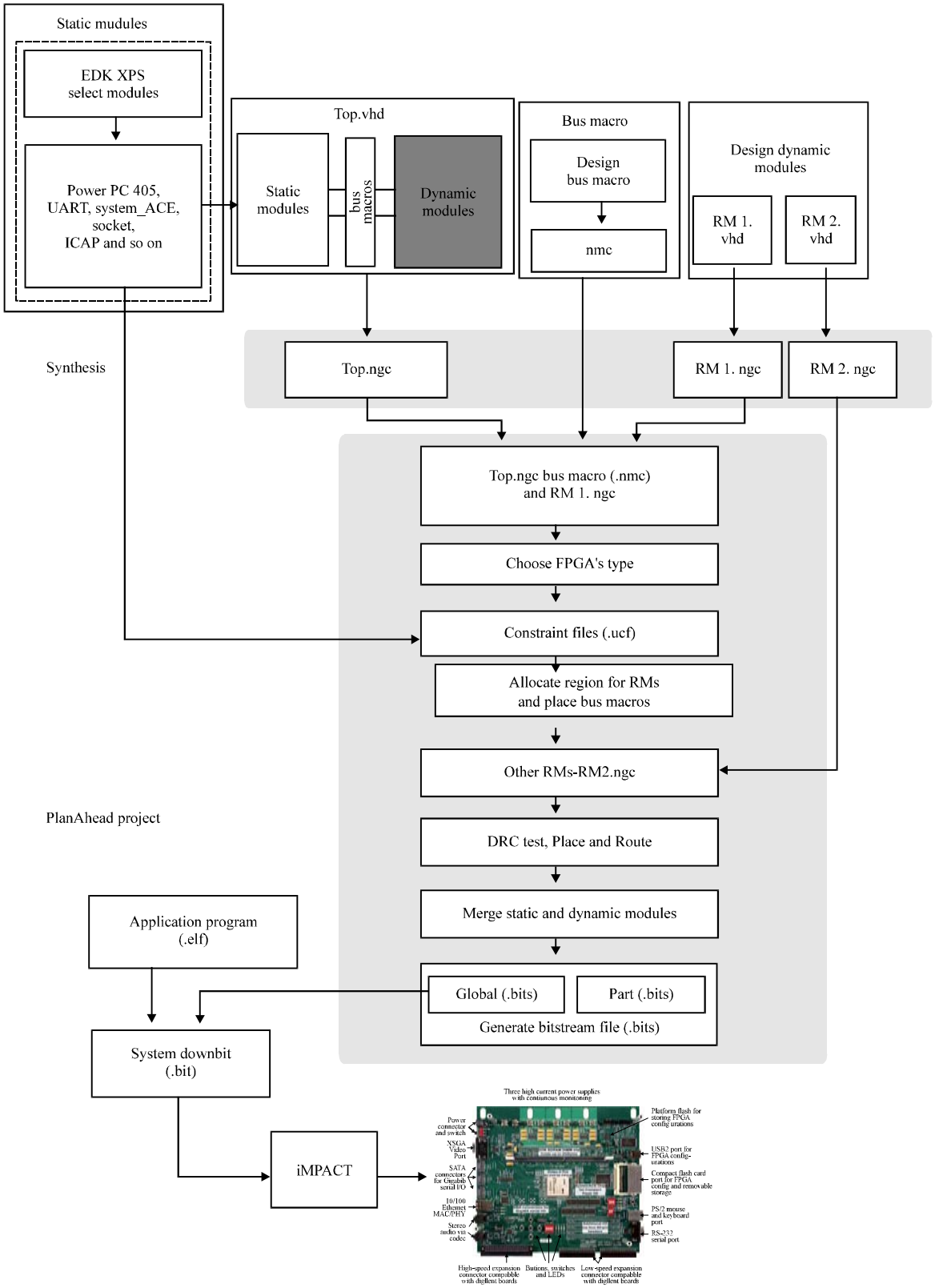

Fig. 4: Design flow of DPR system

Studio that is for hardware platform development and provided by Xilinx. And the processor is the IBM PowerPC hard-core and these peripheral devices are System ACE, RS232, ICAP, Socket and so on.

The reconfiguration operation of the system with the help of static modules is as follows: after serial terminal receives the reconfiguration command, the processor PPC405 follows the command to find the reconfigurable module's bitstream file which is in CF, then write the bitstream data in the BRAM of ICAP with frame as a unit, finally these date are written into the reconfigurable region of FPGA through the ICAP to complete the reconfiguration operation.

Design reconfigurable module: The reconfigurable module which is under the control of the static modules to implement configuration, can achieve new logic function through replacing other modules which lie in dynamic region. Each reconfigurable module achieves specific function and the reconfigurable modules which have the same interface to replace with each other.

One of the advantages of the DPR system is the ability in achieving logic function change in the FPGA's dynamic region which can achieve more functions in a smaller FPGA and reconfigurable module is used to achieve the specific function. Therefore, the reconfigurable module is the core of the system and it can time-share the logic resources of dynamic region and thereby improving the resource utilization. Do pay attention as designing reconfigurable modules: At first, reconfigurable module's resources cannot be greater than the dynamic region's ones. Secondly, the reconfigurable modules which can replace with each other must have the same interface. This study designs two reconfigurable modules to test and verify the DPR system and the two modules are designed as IP that conform the OPB bus standard to ensure the unity of the interface.

Merge modules: Because the static modules and reconfigurable modules are "black box" in the top module, we will use the PlanAhead tool to fill these "black box" to generate a complete DPR system. PlanAhead's main tasks: (1) allocate region and distribute resources for the reconfigurable modules, (2) place the bus macros, (3) place and route, (4) generate bitstream file for each module. Merge the various modules to generate a DPR system as shown in Fig. 4.

EXPERIMENT AND PERFORMANCE ANALYSIS

According to the design flow of the DPR system, we have designed a specific application to implement a DPR system. The application, shown in Fig. 2, with the system has two reconfigurable modules: DES Encryption and DES Decryption (Opencores.org, 2003). We will use Virtex-II Pro Evaluation board to verify the design.

Figure 5 shows the layout of every module of the DRP system and Fig. 6 is the overview of layout of this DPR system and shows that the dynamic region (Fig. 5b, c) is rectangular and the static modules (Fig. 5a) can use the unused routing resources that belong to the dynamic region.

DES algorithm works: At first, the data is encrypted with a key to generate ciphertext data (64 bits) as the output of DES Encryption and then use the same key to decrypt the ciphertext data, restore the plaintext data (64 bits) as the output of DES Decryption. Figure 7 shows the experimental results which consistent with the expectations of the correct result about DES.

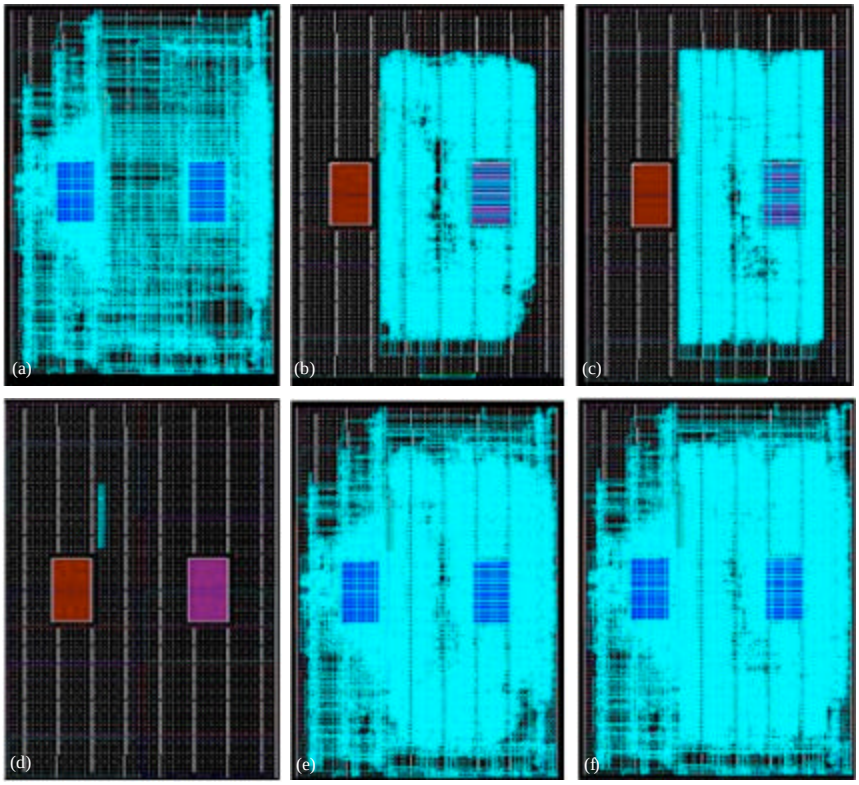

Fig. 5(a-f): Different parts of the Physical layout in DES DPR system, (a) Static module, (b) DES encrypt module, (c) DES decrypt module, (d) Bus macros, (e) DES encrypt system and (f) DES decrypt system

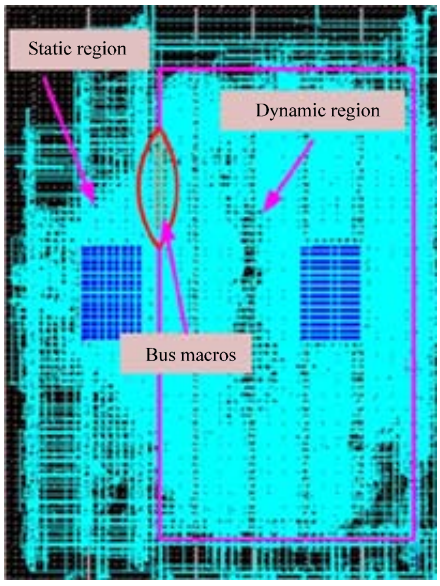

Fig. 6: The layout of the DPR system

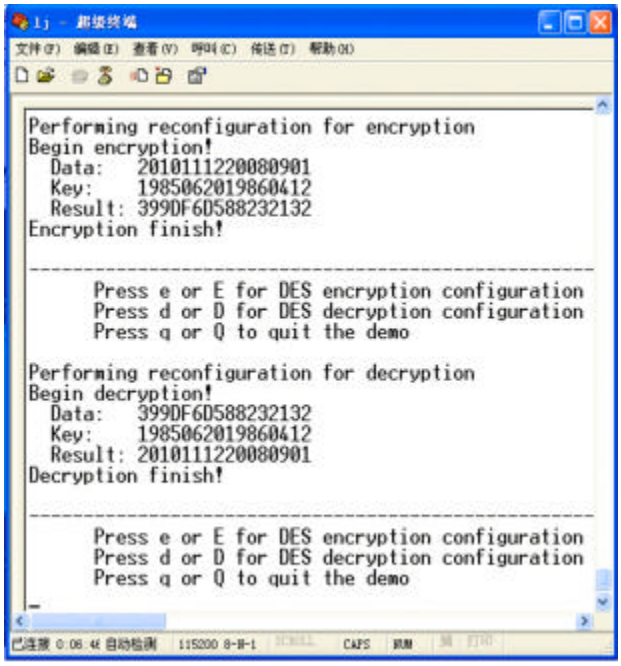

Fig. 7: The results of DES in DPR system

Table 1: Resource usage of the DPR system for DES application

	DES encryption			DES decryption			Static modules			
Resource types	Resource No.	Rate of total resource (%)	Rate of dynamic region's resource (%)	Resource No.	Rate of total resource (%)	Rate of dynamic region's resource (%)	Resource No.	Rate of total resource (%)	Total resource No.	Dynamic region's resource No.
Slices	4,988	36	76	5,382	39	82	1,362	9	13,696	6,508
Slice flip flops	3,982	14	30	4,006	14	30	1,448	5	27,392	13,016
4 input LUTs	5,668	20	43	6,537	23	50	1,513	5	27,392	13,016

Table 1 shows the resources that are used by the DPR system for DES application on Virtex-II Pro XC2VP30 (Xilinx, 2005). The results of Fig. 7 and Table 1 show that encryption module and decryption module can time-share logic resources of dynamic region which greatly saves resources and improves resource utilization. It's worth noting that if more modules share these resources, the resource utilization would be higher.

The DPR system can not only save resources and improve resource utilization, but also save the time of reconfigurable. The time spent on reconfiguration is determined by the following formula:

$$R_t = \frac{B_s}{Clk \times B_w} \quad (1)$$

where, Clk denotes the clock frequency of system, B_w represents the bit wide of transmit data stream and R_t is the value of the size of bitstreams.

Table 2: Bitstream's size and reconfiguration time

Bitstream (.bit)	Size (kb)	Time (sec) (cable freq:6 M)
Static_full.bit	1415	3
Encrypt.bit	533	1
Decrypt.bit	533	1

Table 2 shows the size of the design's bitstream file and the reconstruction time which gets from the iMPACT project's log file. The size of system's bitstream file (static_full.bit) is about 3 times larger than the reconfiguration module's bitsrteam files (encrypt.bit or decrypt.bit) and the reconstruction time is almost 3 times. If the system is dynamic partial reconfiguring, we only download encrypt.bit or decrypt.bit, the time only spent 1 sec. If the system is global reconfiguration, both encrypt.bit and decrypt.bit are needed to load into FPGA, the time spent is 2 sec. Hence, dynamic partial reconfiguring can save 50% reconfiguration time in compared with global reconfiguration. In a word, the DPR system can greatly save reconstruction time and save storage space.

CONCLUSION

In this study, we have studied the design method, discussed the DPR system's structure and composition, studied the design flow of the DPR system and finally implement an EAPR based DPR system with the help of PlanAhead tool. The DPR system for DES encryption and decryption has been implemented and validated on the Virtex-II Pro XC2VP30 development platform. The experimental results show the correctness of the system designed following the design flow. Performance of the experiment shows that the system greatly improves FPGA's resource utilization and saves reconfigurable time.

ACKNOWLEDGMENTS

This study is supported by the National High Technology Research and Development Program ("863"Program) of China Foundation of China under Grant No. 2007AA01Z104, the Natural Science Foundation of Hunan Province under Grant No. 07JJ6136, the Fundamental Research Funds for the Central Universities. Exceedingly the authors want to thank Xilinx for providing development boards.

REFERENCES

Becker, J., M. Hubner, G. Hettich, R. Constapel, J. Eisenmann and J. Luka, 2007. Dynamic and partial FPGA exploitation. Proc. IEEE, 95: 438-452.

Bobda, C., 2007. Introduction to Reconfigurable Computing: Architectures, Algorithms and Applications. 1st Edn, Springer Publishing Company Inc., New York, USA., ISBN-13: 9781402060885, Pages: 359.

Claus, C., F.H. Muller, J. Zeppenfeld and W. Stechele, 2007. A new framework to accelerate Virtex-II Pro dynamic partial self-reconfiguration. Proceedings of the IEEE International Parallel and Distributed Processing Symposium, March 26-30, 2007, Long Beach, CA., USA., pp: 1-7.

Compton, K. and S. Hauck, 2002. Reconfigurable computing: A survey of systems and software. ACM Comput. Surveys, 34: 171-210.

Corbetta, S., M. Morandi, M. Novati, M.D. Santambrogio, D. Sciuto and P. Spoletini, 2009. Internal and external bitstream relocation for partial dynamic reconfiguration. IEEE Trans. Very Large Scale Integr. Syst., 17: 1650-1654.

Li, Y., P.P. Gu and X.S. Wang, 2011. The implementation of semaphore management in hardware real time operating system. Inform. Technol. J., 10: 158-163.

McDonald, E.J., 2008. Runtime FPGA partial reconfiguration. Proceedings of the IEEE Aerospace Conference, March 1-8, 2008, Big Sky, MT., USA., pp: 1-7.

Opencores.org, 2003. Basic DES crypto core: Overview. http://opencores.org/project,basicdes

Papadimitriou, K., A. Anyfantis and A. Dollas, 2010. An effective framework to evaluate dynamic partial reconfiguration in FPGA systems. IEEE Trans. Instrum. Meas., 59: 1642-1651.

Parris, M.G., 2008. Optimizing dynamic logic realizations for partial reconfiguration of field programmable gate arrays. M.S. Thesis, B.S. University of Louisville, Louisville, KY., USA.

Peiravi, A. and M.J. Rahimzadeh, 2009. Design and implementation of a novel high performance content processor for storage disks using an exact string matching architecture. J. Applied Sci., 9: 488-496.

Ramadass, N., S. Natarajan and J.R.P. Perinbam, 2007. Dynamically reconfigurable (Self-modifiable) architecture for embedded system-on-chip applications. Inform. Technol. J., 6: 66-73.

Ramakrishnan, M. and S. Shanmugavel, 2007. Hardware implementation of TORA protocol in mobile Ad-hoc network node. Inform. Technol. J., 6: 345-352.

Wu, Z., N. Zhao, S. Li and G. Ren, 2009. A novel PCM/FM multi-symbol detection algorithm for FPGA implementation. Inform. Technol. J., 8: 583-588.

Xilinx, 2004. Two flows for partial reconfiguration: Module based or difference based. Xilinx Application Note XAPP290 (V1.2), Xilinx Inc., San Jose, CA., USA.

Xilinx, 2005. Virtex-II Pro and Virtex-II ProX FPGA user guide. Xilinx Application Note UG012 (v4.1), Xilinx Inc., San Jose, CA., USA.

Xilinx, 2006. Early access partial reconfiguration user guide. http://www12.informatik.uni-erlangen.de/esmwiki/images/f/f3/Pr_flow.pdf